OPTIMIZATION OF UTD DINING SERVICES

**  
Photograph - Flickr:** [**https://www.flickr.com/photos/wadegriffith/15533073810**](https://www.flickr.com/photos/wadegriffith/15533073810)

***Sriram Kumar Reddy Challa - sxc190047***

***Namratha Nagaraj - nxn180027***

***Manali Thorat - mmt190000***

***Ritu Mamtani - rxm180084***

***Dilip Merala - dvm170000***

Table of Contents

[Project Video Link : 4](#_Toc26556955)

[Executive Summary 4](#_Toc26556956)

[Problem Statement 5](#_Toc26556957)

[Business Process Model using BPMN 6](#_Toc26556958)

[Use Case Diagram 9](#_Toc26556959)

[Use Case Name: Login 10](#_Toc26556960)

[Use Case Name: Order food 10](#_Toc26556961)

[Use Case Name: Request for takeaway 11](#_Toc26556962)

[Use Case Name: Code generation 12](#_Toc26556963)

[Use Case Name: Give code to cashier 13](#_Toc26556964)

[Use Case Name: Predict quantity 14](#_Toc26556965)

[Use Case Name: Check for available items 14](#_Toc26556966)

[Use Case Name: Gets the specific vendor 15](#_Toc26556967)

[Use Case Name: Check cutlery after regular intervals 15](#_Toc26556968)

[Use Case Name: Notify waiter if the cutlery weight is below threshold 16](#_Toc26556969)

[1. Check the availability of the cutlery 16](#_Toc26556970)

[2. Notify the waiter 16](#_Toc26556971)

[Use Case Name: Refill stock 16](#_Toc26556972)

[Use Case Name: Send Approval for quantity required 17](#_Toc26556973)

[Use Case Name: Order products from vendor 17](#_Toc26556974)

[Use Case Name: Receive products from vendor. 18](#_Toc26556975)

[Data Dictionary 19](#_Toc26556976)

[Class diagram without methods 20](#_Toc26556977)

[Complete Class Diagram with methods 20](#_Toc26556978)

[Sequence Diagram 21](#_Toc26556979)

[Functional Specification Document for the proposed system 23](#_Toc26556980)

[Interface Design 24](#_Toc26556981)

[Database design 26](#_Toc26556982)

[Entity- Relationship Diagram 27](#_Toc26556983)

[Software Design 28](#_Toc26556984)

[Project Management Deliverables 35](#_Toc26556985)

[1. Project Activities 35](#_Toc26556986)

[3. Allocation of activities to team members 35](#_Toc26556987)

[3. Planned timeline 36](#_Toc26556988)

[4. Execution timeline 36](#_Toc26556989)

[5. Minutes of project meetings (date, time, team members present, topic discussed) 37](#_Toc26556990)

# Project Video Link :

<https://youtu.be/ReVSWSGWLQM>

# Executive Summary

Food is either lost or wasted throughout the supply chain from initial agricultural production all the way to final consumption. Food that never gets eaten also represents a waste of resources, such as land, water, energy, soil, seeds , most of all energy, and other inputs used in its production, increasing green gas emissions in vain.

According to the Food Recovery Network, irresponsible college campus dining methodologies lead to over 22 million pounds of food waste yearly. It is not merely the fault of students but of the dining halls too. This is because dining facilities do not always accurately predict the amount of food students will consume, leading to large amounts of waste everyday. Due to health regulations, even if a piece of food has barely been touched, the dining hall cannot legally donate it to a food bank. Unnecessary food wastage in America, specifically on college campuses, is a large and often-ignored issue.

The University of Texas at Dallas (UTD) is a rapidly-growing institution. According to a recent News article by the UTD news center, student enrollment has grown more than nine percent in the past year (The University of Texas at Dallas, Influx of New Students Pushes Fall Enrollment Past 23,000 Mark). So the number of students enrolling for the meal plan at the dining hall proportionally increases, which shall also increase food wastage if not optimized. We believe that we have devised an Operation Plan that can help us minimize food wastage behind the counter.

This project is intended towards proposing certain methods to optimize the dining process. If we can predict the count of student turnout for a meal, then we can reduce food wastage behind the counter. To implement this, we can use Machine Learning techniques for prediction. Also, based on the consumption rate of each item on the menu, we can optimize the Dining inventory management system. Based on consumption rate and food expiry date, we can predict the quantity of food to be ordered or kept in store.

A digital platform like a website can facilitate the food-ordering and delivery process. Social media can be used to promote food wastage awareness among students by sharing posts, updating daily food wastage in the dining area. We can also get reviews on the quality of food served, and menu and sanitation of the dining hall. Using this platform, the Takeaway plan can be made more student-friendly, so that lucrative offers can be extended to students and leftover food at EOD is reduced.

# Problem Statement

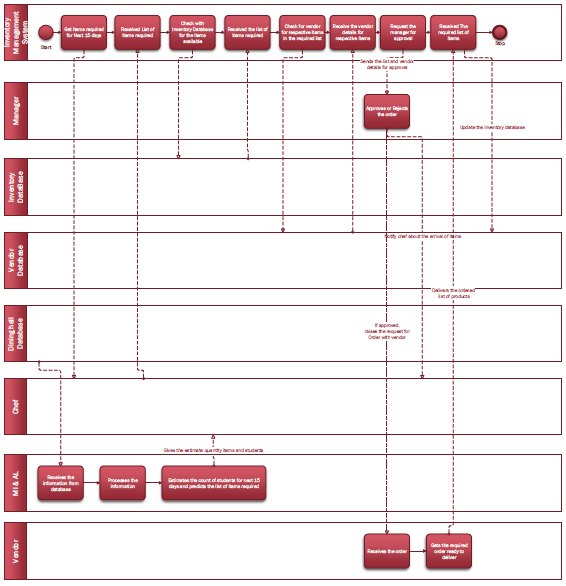
1. Optimize dining hall functionalities for reduction of food wastage by implementing Machine Learning techniques to predict student turnout for each meal and improving Inventory Management to place orders to vendors, manage food items according to their expiry dates and other factors.
2. a. Need to upgrade current sick meal plan to a TakeAway plan through a convenient digital interface with secure login process for students to order food online, pick up their TakeAway meals, and provide continuous feedback thus both benefiting students and increasing sales.  
   b. Increase food-wastage awareness through a social media platform.
3. During peak time, there is a shortage of cutlery in the student Dining Halls. Currently, there are three stands with racks for spoons, forks and knives, located at different parts in the dining hall. We need an IoT based solution to automatically detect shortage of cutlery and notify the waiter. This will be implemented by placing a weight sensor under each rack, which sends a signal to notify for a refill of specified cutlery at that particular location in the dining hall.

# Business Process Model using BPMN

This section presents solutions to the above problem statements in the form of BPMN diagrams for the inventory management system, food-ordering website, and cutlery replacement using IoT.

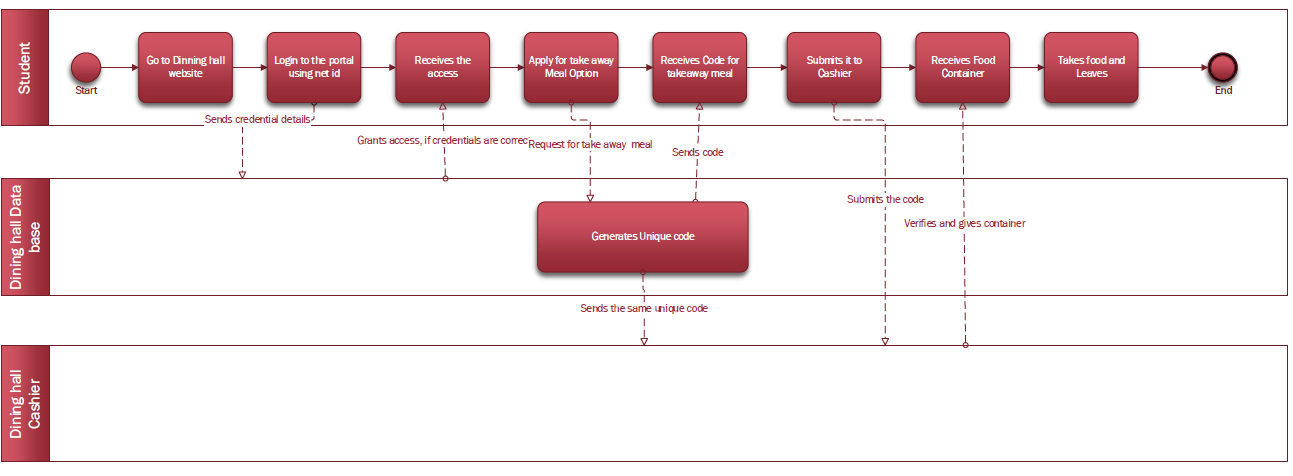
**Inventory Management**

This BPMN displays the coordination of Order Management systems between Kitchen, Storage, Manager and Vendor.



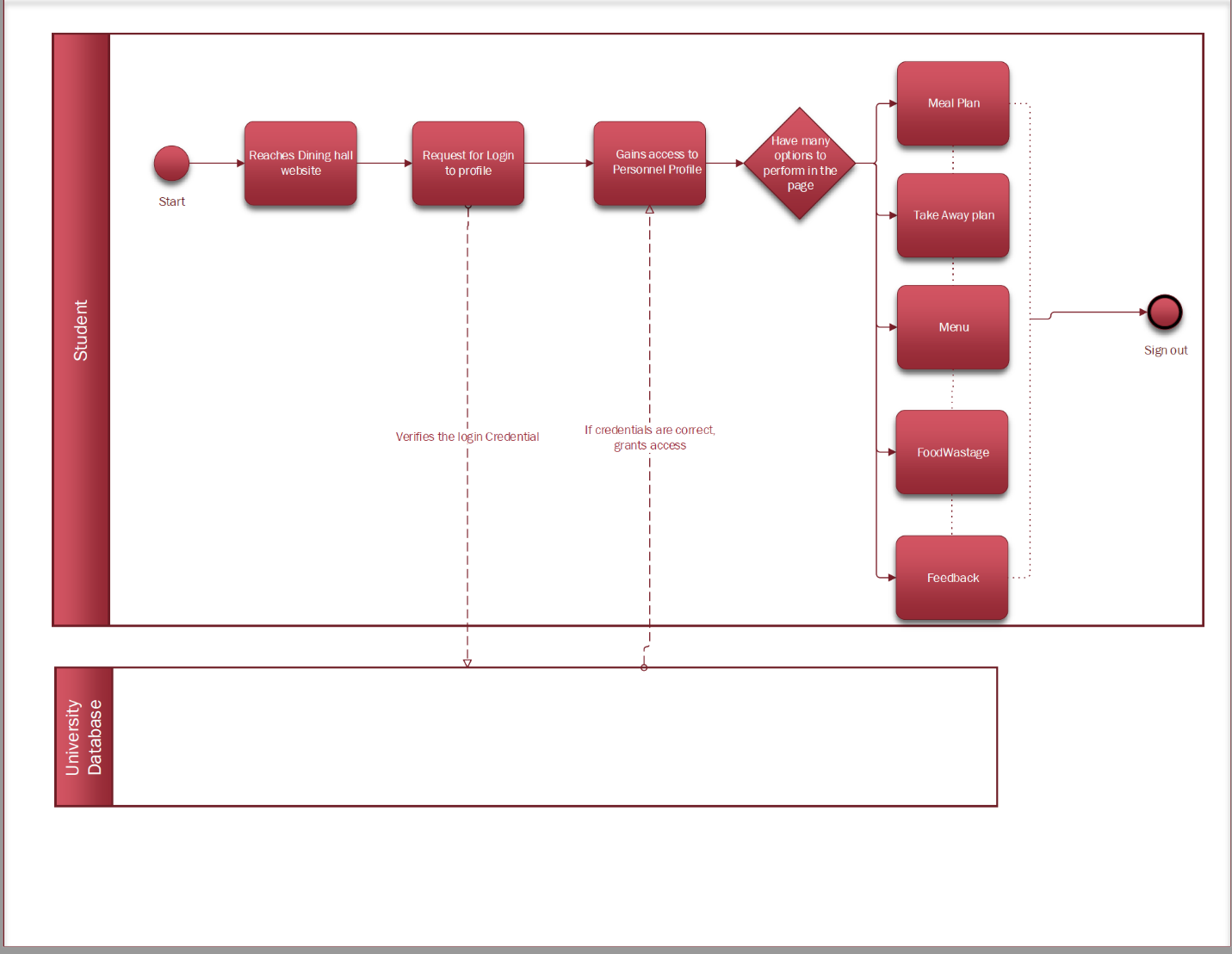
TakeAway Plan

This BPMN displays the process of our digital interface to enable students to place orders and TakeAway food.



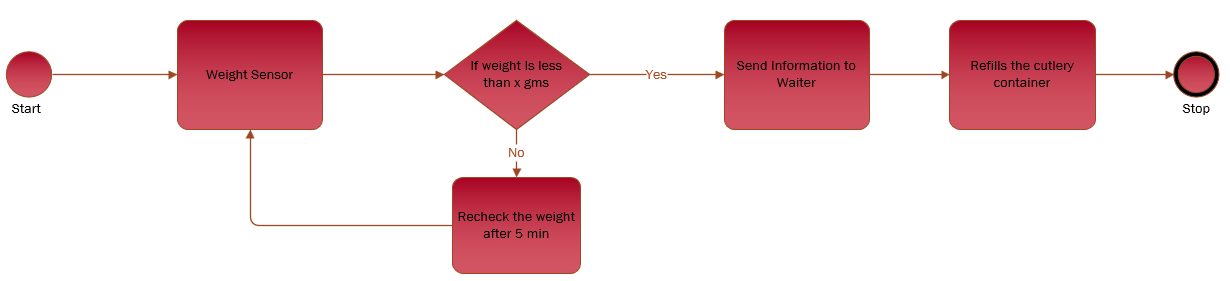
Website

Front End interface for Student to engage with the Dining Services



Cutlery

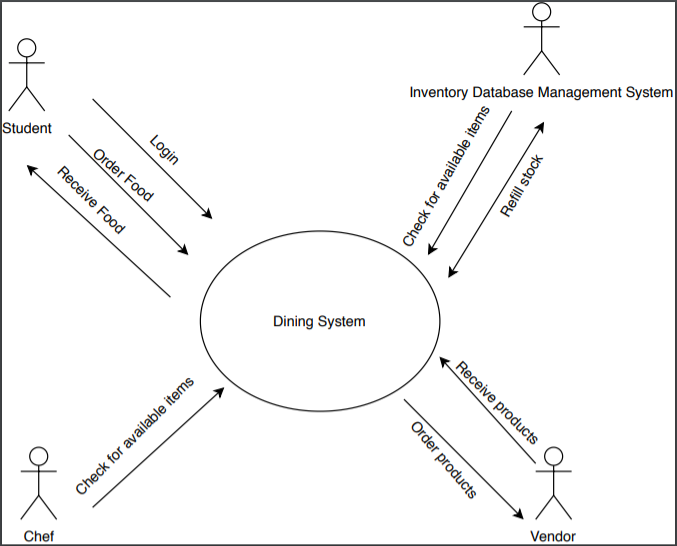
To monitor shortage of cutlery at different locations of Dining Hall



## 

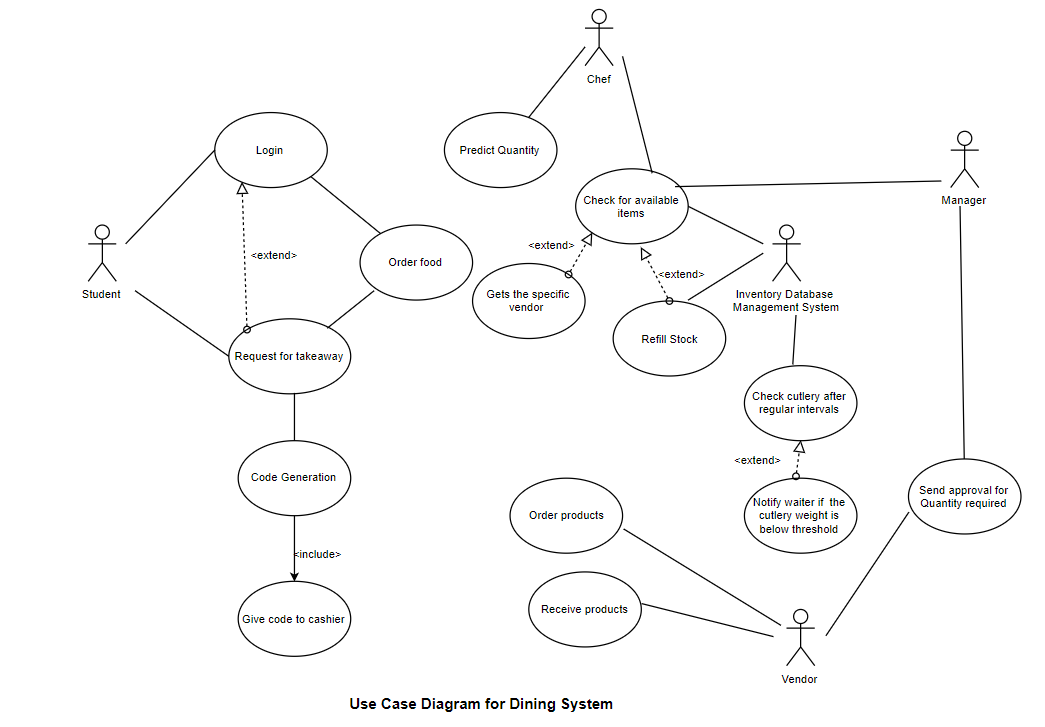
Context Diagram

Here is a diagrammatic representation of the Dining system and its services showing actors that interact with the Centralized Dining System.



# Use Case Diagram

Here is a concise model of our solutions and the involved components:



Use Case Descriptions

## Use Case Name: Login

**Primary Actor:** Student

**Stakeholders:** Student

**Description:** Load student profile

**Trigger:** User logs into the system with UTD-ID and password

**Relationships:** Website Database

**Includes:**

**Extends:**

**Normal flow of events:**

1. Login information is submitted.
2. System sends username and password to UTD system for authentication and retrieving student profile.
3. Order request is received by the system.

**Exceptional Flow:**

Authentication failed, display invalid username and/or password (error screen)

## Use Case Name: Order food

**Primary Actor:** Student

**Stakeholders:** Student

**Description:** Student can order food after logging in the system.

**Trigger: After login, the menu page opens.**

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:**

1. After login
2. Food order request is sent.

## Use Case Name: Request for takeaway

**Primary Actor:** Student

**Stakeholders:** Student

**Description:** Student can order takeaway after logging in the system.

**Trigger:** After selecting takeaway meal, a code is generated and sent to students. He then submits it to the cashier to order food.

**Relationships:**

**Includes:**

**Extends: Login**

**Normal flow of events:**

1. After login, student can request for takeaway meal.
2. A code gets generated.
3. This code should be submitted to cashier so that the takeaway meal order gets confirmed.
4. After receiving code, cashier generates invoice.
5. Food is prepared and delivered.

**Exceptional Flow:**

If the code is not received then order is not confirmed.

## Use Case Name: Code generation

**Primary Actor:** Student

**Stakeholders:** Student

**Description:** After ordering takeaway meal,Student gets a code which needs to be sent to the cashier for order confirmation.

**Trigger:** After selecting takeaway meal, a code is generated and sent to students. The student submits it to the cashier for ordering food.

**Relationships:**

**Includes: Give code to cashier**

**Extends:**

**Normal flow of events:**

1. After login student can request for takeaway.
2. A code gets generated.
3. This code should be submitted to cashier so that the takeaway meal order gets confirmed.
4. After receiving code, cashier generates invoice.
5. Food is prepared and delivered.

**Exceptional Flow:**

If the code is not received then order is not confirmed. Error is generated.

If the code is not generated, request again for the code.

## Use Case Name: Give code to cashier

**Primary Actor:** Student

**Stakeholders:** Student

**Description:** After getting the code, send it to the cashier.

**Trigger:** After selecting takeaway meal, a code is generated and sent to students. The student submits it to the cashier for ordering food.

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:**

1. After login student can request for takeaway meal.
2. A code gets generated.
3. This code should be submitted to cashier so that the takeaway meal order gets confirmed.
4. After receiving code, cashier generates invoice.
5. Food is prepared and delivered.

**Exceptional Flow:**

If the code is not received then order is not confirmed.

## Use Case Name: Predict quantity

**Primary Actor:** Chef

**Stakeholders:** Chef

**Description:** Chef predicts the quantity of the products required in the inventory.

**Trigger: Chef checks the inventory**

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:**

1. Chef checks the inventory

2. Predicts the quantity of products required.

## Use Case Name: Check for available items

**Primary Actor:** Chef, Inventory database management system

**Stakeholders:** Chef, Inventory database management system

**Description:** The items in the inventory are checked for their quantity required.

**Trigger:** Items are checked after a certain time interval for availability.

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:**

1. Items are checked after a certain time interval for availability.

## Use Case Name: Gets the specific vendor

**Primary Actor:** Inventory database management system

**Stakeholders:** Inventory database management system

**Description:** Get data about the specific vendor to order items

**Trigger:** Get data about the specific vendor to order items

**Relationships:**

**Includes:**

**Extends:** Check for available items

**Normal flow of events:**

1. Check the availability of products

2. Select the specific vendor

## Use Case Name: Check cutlery after regular intervals

**Primary Actor:** Inventory database management system

**Stakeholders:** Inventory database management system

**Description:** Get data about cutlery weight after regular intervals.

**Trigger:** Check cutlery weight after regular intervals

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:**

1. Check the availability of cutlery

2. Refill if the weight is below threshold

## Use Case Name: Notify waiter if the cutlery weight is below threshold

**Primary Actor:** Inventory database management system

**Stakeholders:** Inventory database management system

**Description:** After checking the cutlery weight at regular intervals, if the weight is below threshold then notify the waiter so that it can be restocked.

**Trigger:** If weight< threshold

then notify waiter

**Relationships:**

**Includes:**

**Extends:** Check cutlery after regular intervals

**Normal flow of events:**

## 1. Check the availability of the cutlery

## 2. Notify the waiter

## Use Case Name: Refill stock

**Primary Actor:** Inventory database management system

**Stakeholders:** Inventory database management system

**Description:** If the quantity is less then refill the stocks.

**Trigger:** If the quantity is less then refill the stocks.

**Relationships:**

**Includes:**

**Extends:** Check for available items

**Normal flow of events:**

1. Check the availability of products

2. If the quantity is less then refill the stocks.

## Use Case Name: Send Approval for quantity required

**Primary Actor:** Manager

**Stakeholders:** Manager

**Description:** Approve or reject the quantity to be ordered

**Trigger:** If food availability is less, order items from the vendor and approve the order request

**Relationships:**

**Includes:**

**Extends:**

**Normal flow of events:** Order products from the vendor

**Exceptional Flow:** If the manager does not find the order request to be correct, he rejects the order request.

## Use Case Name: Order products from vendor

**Primary Actor:** Vendor

**Stakeholders:** Vendor

**Description:** Order products from vendor

**Trigger:** If food availability is less, order items from the vendor

**Relationships:**

**Includes:**

**Extends: Login**

**Normal flow of events:** Order products from the vendor

## Use Case Name: Receive products from vendor.

**Primary Actor:** Vendor

**Stakeholders:** Vendor

**Description:** Once the order is complete and is ready, receive from the vendor.

**Trigger**: Receive the products from vendor after the order is complete.

**Relationships:**

**Includes:**

**Extends: Login**

**Normal flow of events:**

Receive the products from vendor after the order is complete.

# Data Dictionary

Login info = Username + password

username = Net ID

password = data element

Authorize Log In

Email Address: Data Element

Password: Data Element

User Info = First Name + Last Name + Email Address + Password + ZipCode + Birthday + Gender

Order food = Login + Meal plan + Display menu

Meal plan = [Normal | Takeaway]

Display Menu = Menu Id + Menu Type + Item No + Item Name + Item Type + Item Price

Request for TakeAway meal = login + code + cashier information + order confirmation + invoice

Login info = Username + password

Code generation = Unique code number

Cashier information = order confirmation + food delivery

Invoice = Billing No + Amount + Billed To + Billing Date + City + State + Zip code + Email ID

Payment information = payment method + invoice

Predict quantity = {Quantity consumed + Quantity received}

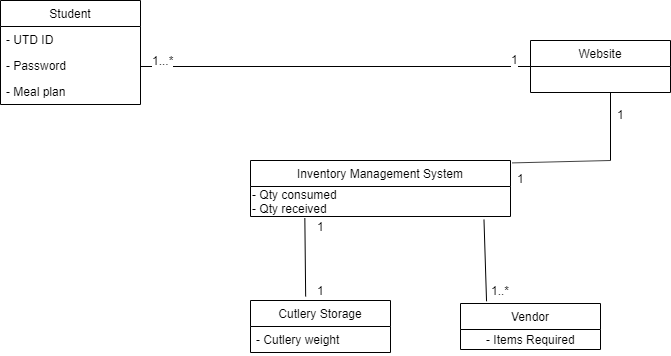
Check for available items = Predict quantity + Refill stock

Order products from vendor = items required + process delivery status + vendor data

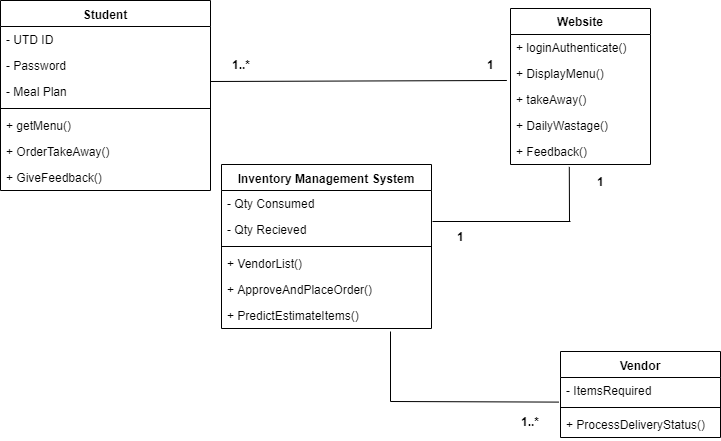
Vendor data = vendor name + vendor contact + items provided + vendor location

Receive products from vendor = Make payment + Refill stock

# Class diagram without methods

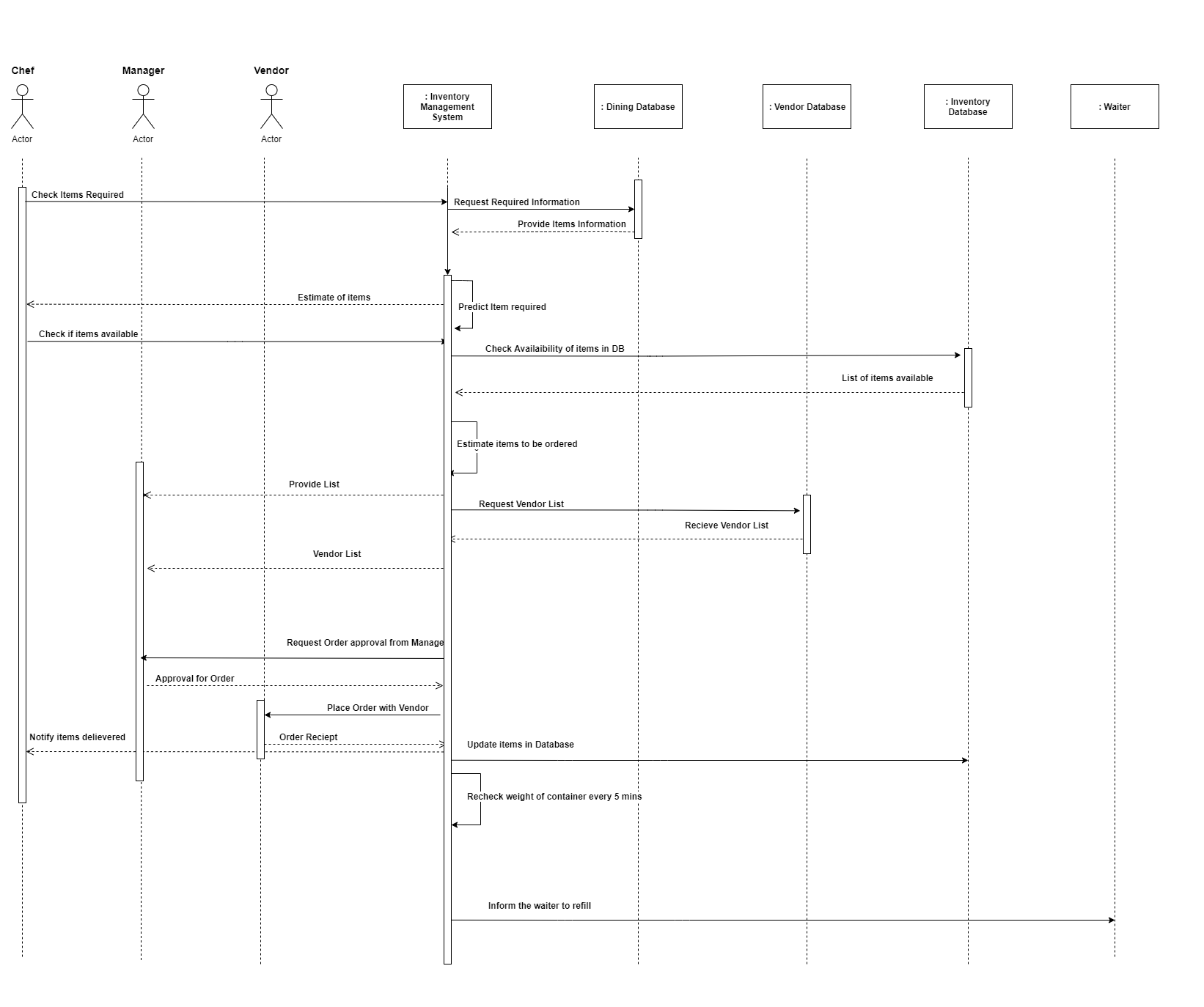


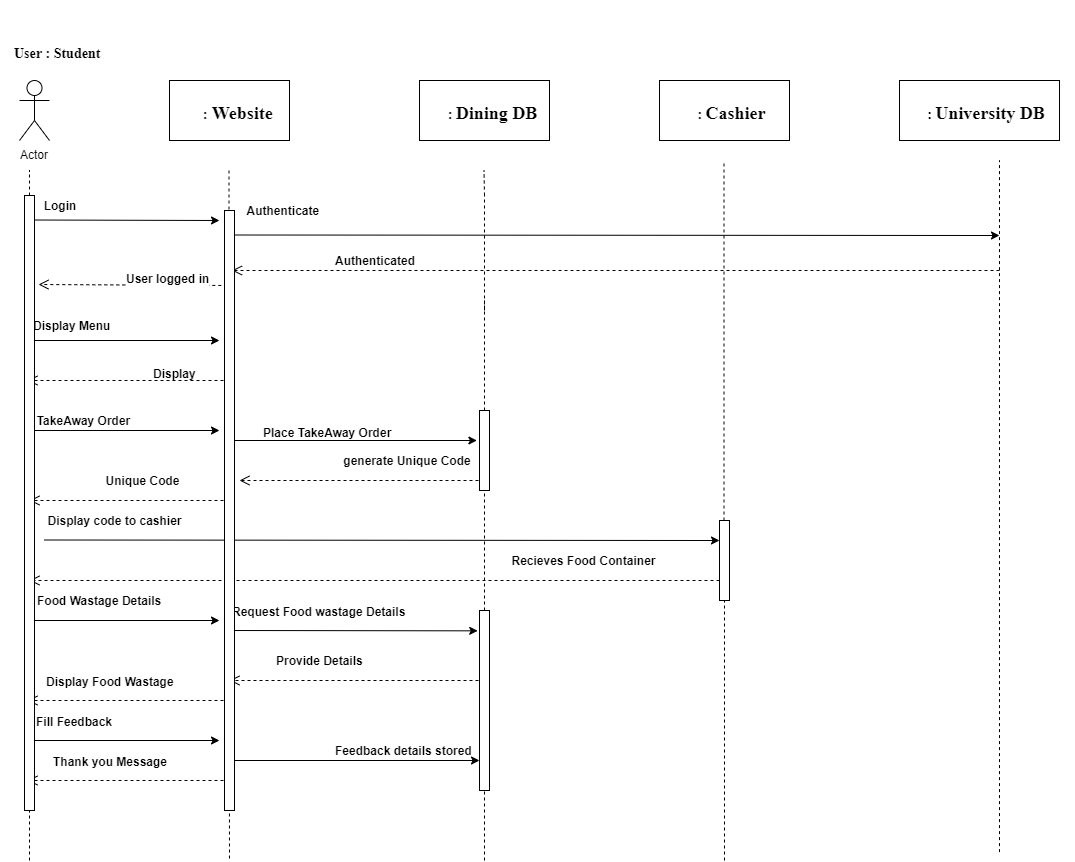
# Complete Class Diagram with methods



# Sequence Diagram

This sequence diagram shows the flow of events in the system and the order in which each task takes place.





# Functional Specification Document for the proposed system

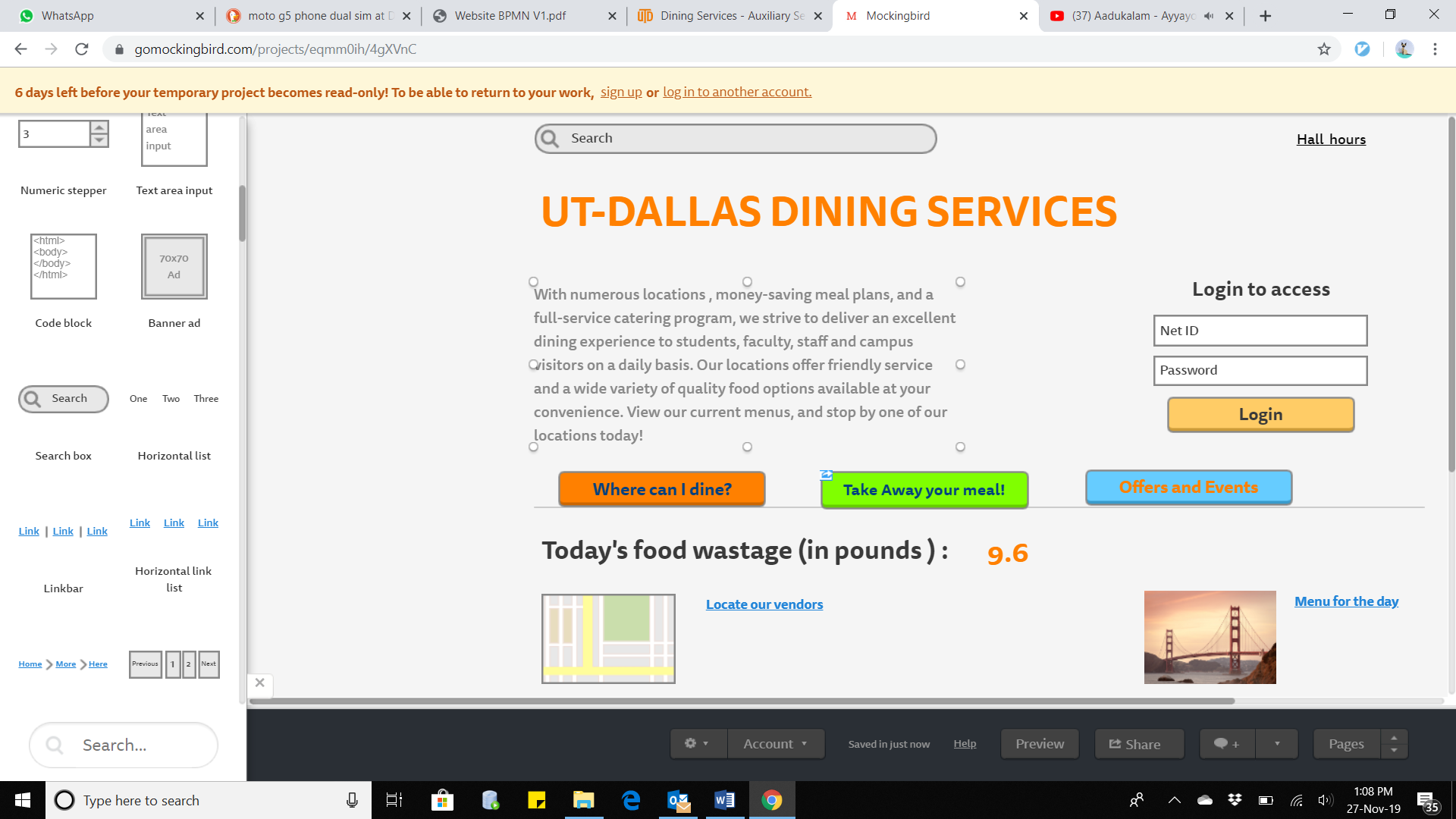
• The proposed system will optimize inventory management by predicting the items required, automating the order process, and help in reducing food wastage by implementing techniques to predict student turnout for each meal. Here, Machine learning is used to gather data on how often an item is ordered from the inventory, and how much of it is ordered. These values can be compared with how many items were actually sold, and analysis will be performed on whether most of them were sold, or disposed due to reasons like low shelf life. This will enable us to predict future purchases more carefully.

•The dining website will help in optimizing the TakeAway plan and storing order history for easy selection. In the TakeAway plan, the student order on the website and a unique code gets generated for each order. This code should be shown to the waiter for receiving the order. A social media platform should be developed to keep students updated about meal plans and daily menu, improve awareness about food wastage, and get continuous feedback about the food served.

• This system also aims to handle the availability of cutlery, wherein every cutlery storage container will have sensors. Every 5 minutes, the system checks if the weight is above the predetermined threshold. If not, it will notify the waiter to fill the cutlery container.

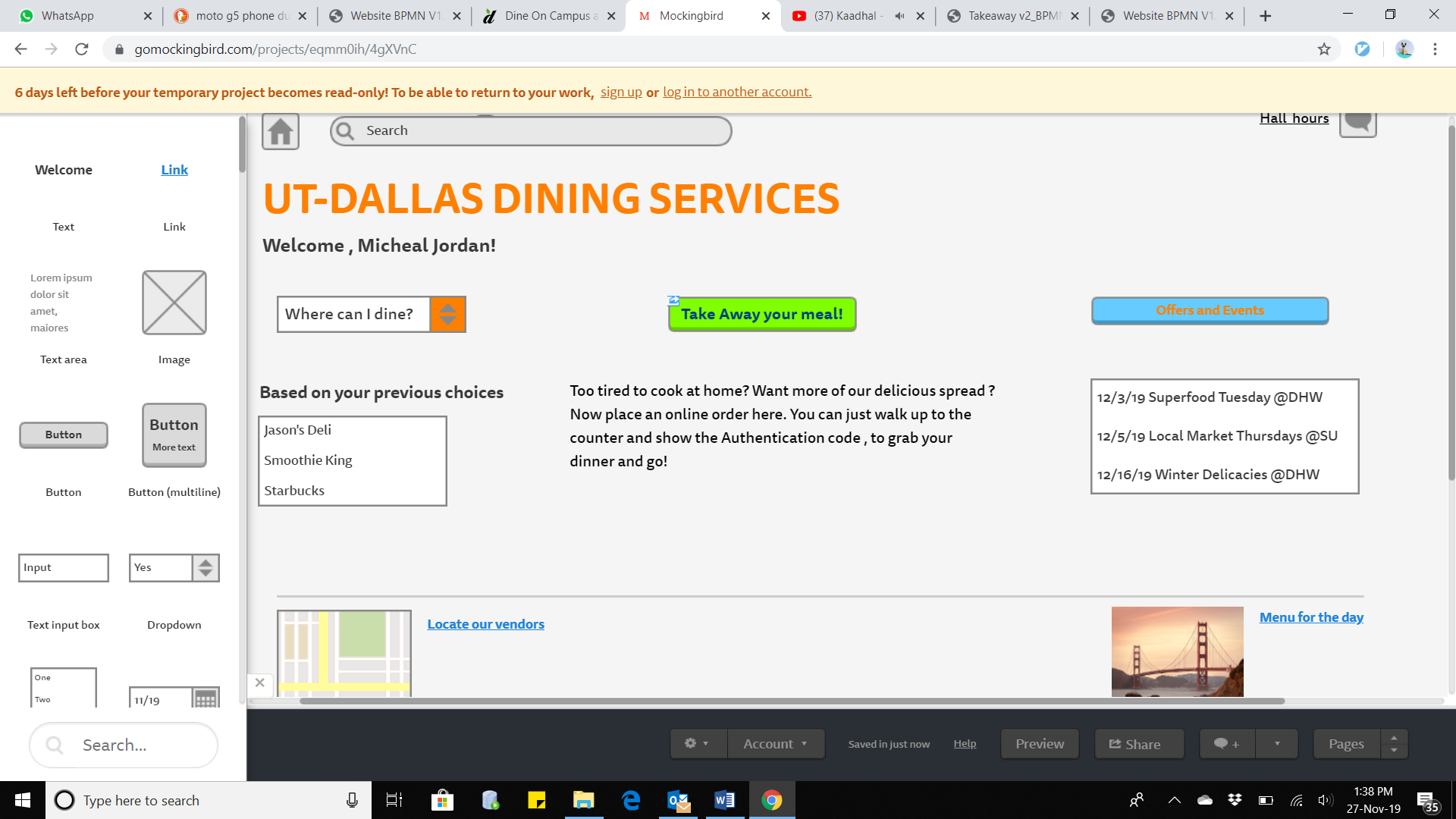
# Interface Design

**Main Login Page**



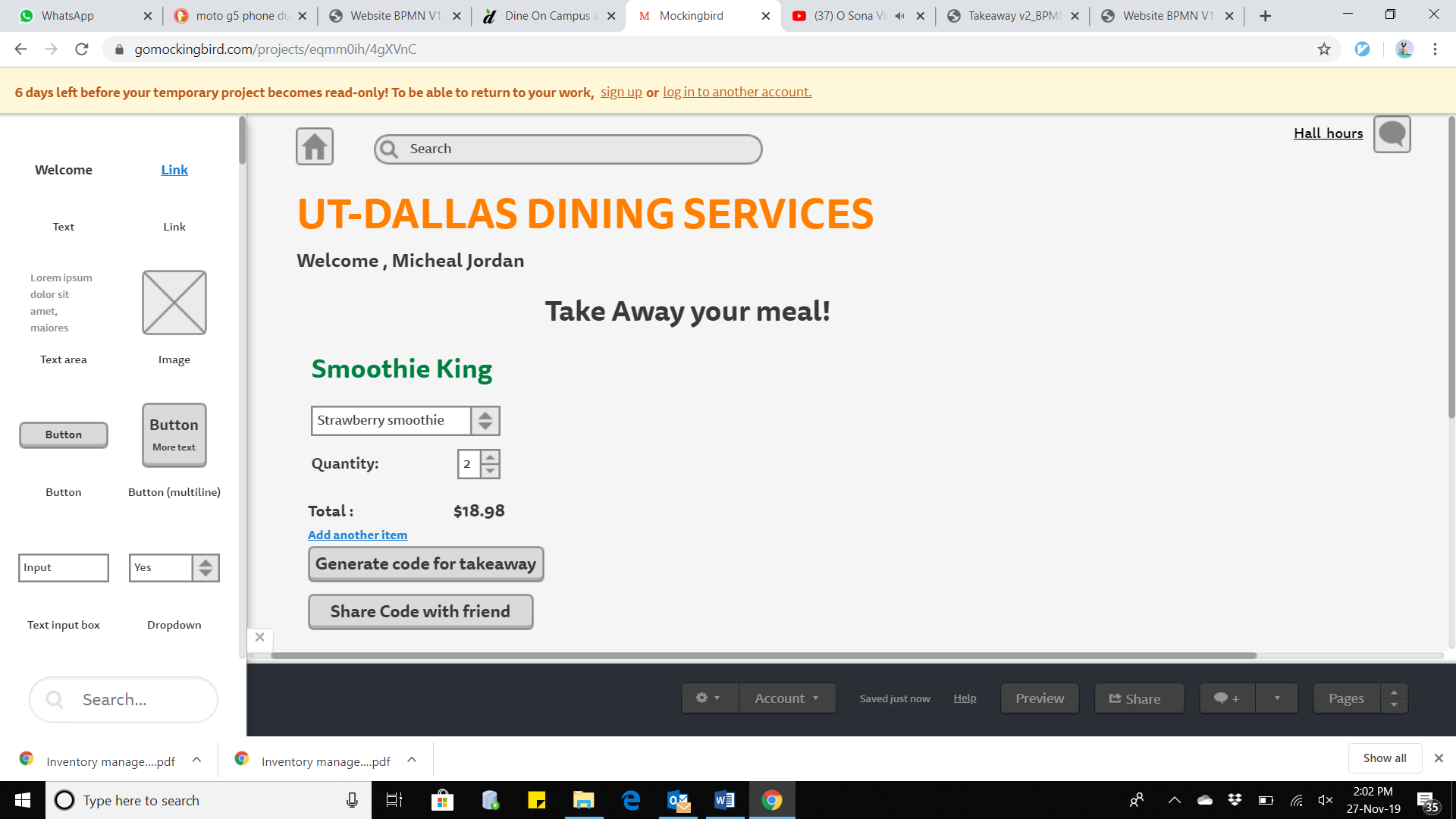
The UT-Dallas Dining services includes secure login. We have new tabs through which we can see the various outlets we can eat at, utilize the option of having a TakeAway meal, and also get updated on events and offers conducted by Dining. We also display food wastage right here so that awareness is maximum.

**Meal order page**

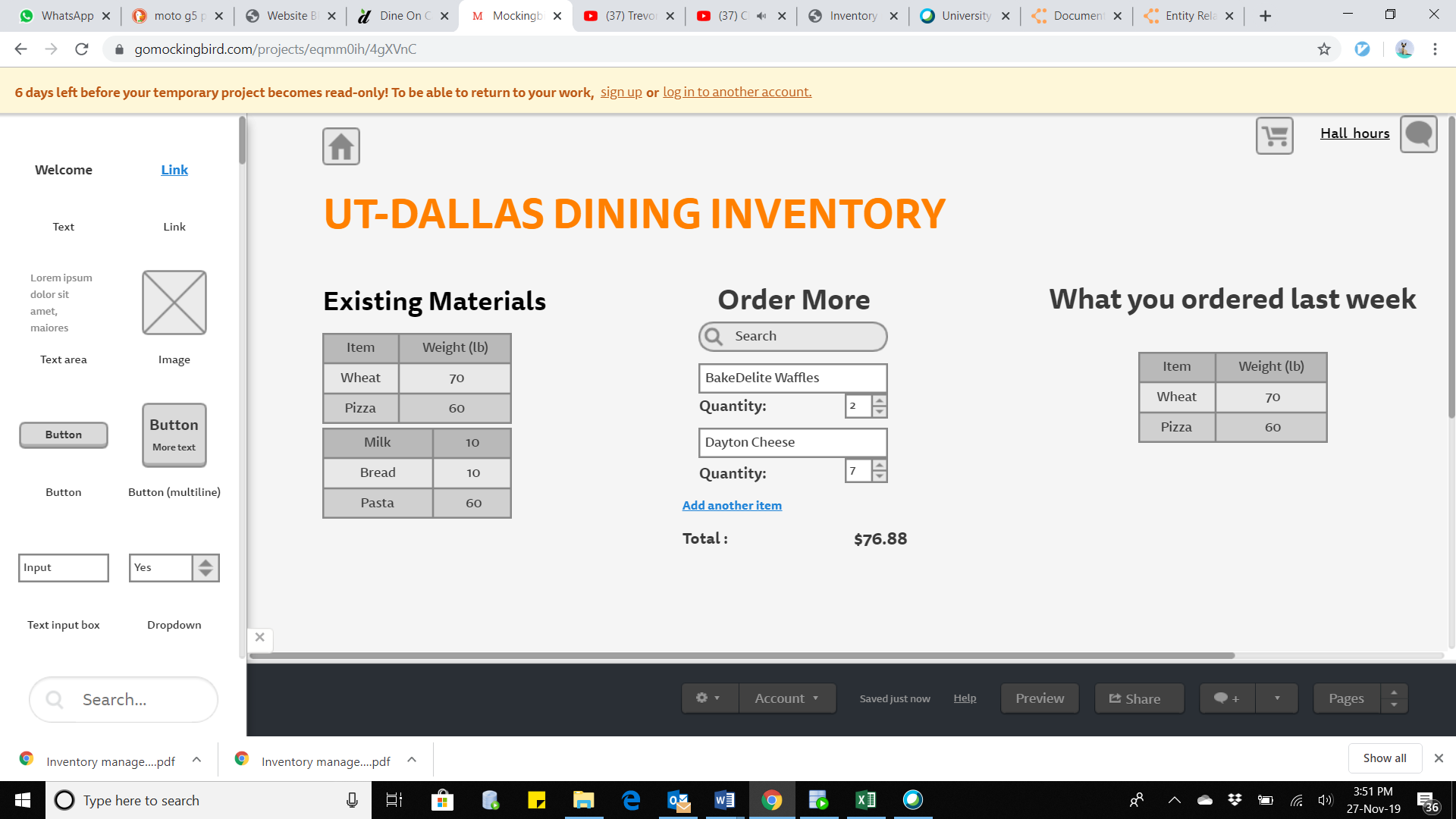


Once a student logs in, he can view his meal plan , the menu for the day etc. This page also provides the TakeAway option.

**Meal Receipt Page**



**Inventory Status Page**



The website when accessed by Dining Service personnel like Manager , displays an interface for them to place orders to vendors.

# Database design

Index :

Bold - Primary Keys

Italics - Foreign Keys

1. Overall UTD Inventory ( ***Order ID, Item No*** , Vendor ID, Item Date Ordered , Item Date Delivered , Item Date Expiring , Item Quantity Ordered )

OrderID is non null and unique, and must exist in Cashier table.

1. VendorData ( Item Ordered , ***Item No***., Item Date Ordered , Date Delivered , Quantity)

Item No. is unique and non-null.

Item No. must exist in table Overall Inventory.

1. Cashier (***Order ID***, OrderDate, Date Sold , Quantity Required , Receipt No.)

UTD ID must exist in table Login.

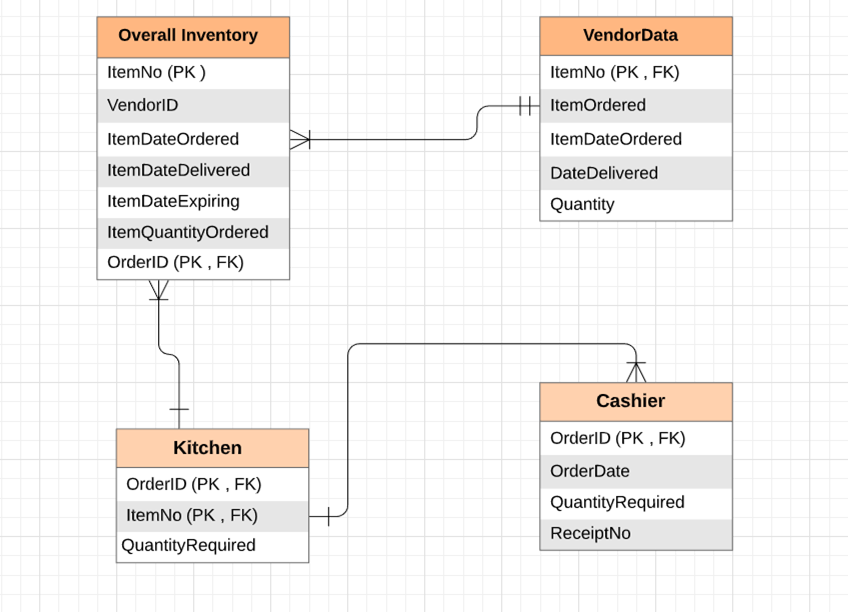
Order ID is non-null.

1. Kitchen( ***Order ID, Item No*** , Quantity Required)

Order ID is unique and non-null.

Item No. is unique and non-null.

# Entity- Relationship Diagram



# Software Design

|  |  |  |
| --- | --- | --- |
| **Method Name:** Login | **Class Name:** Website | **ID:** 1 |
| **Clients (Students)** | Main method | |
| **Associated Use Case** | Login | |
| **Description of Responsibilities** | 1. Get user details  2. Call authenticate method  3. If login successful, display Menu  4. If login unsuccessful, display error | |
| **Arguments Received** | User login and password | |
| **Type of value returned** | Menu | |
| **Pre-conditions** | None | |
| **Post-conditions** | Error or Menu is displayed | |

**Pseudocode**:

Receive login details from user

Send details to DB using authenticate()

IF user is authenticated, then display menu

ELSE display error and ask to create new login

|  |  |  |
| --- | --- | --- |
| **Method Name:** TakeAwayOrder | **Class Name:** Website | **ID:** 2 |
| **Clients (Students)** | Main method | |
| **Associated Use Case** | Order Food | |
| **Description of Responsibilities** | 1. Get order from authenticated user 2. Send the order to dining DB 3. Generate unique code at DB 4. If code not generated, display error prompt to try again 5. If code generated, display code to user | |
| **Arguments Received** | Food order | |
| **Type of value returned** | Unique Code or try again prompt | |
| **Pre-conditions** | None | |
| **Post-conditions** | A unique code is displayed | |

**Pseudocode**:

Receive food order from student/user

Send order to Dining DB

Send request to generate code

IF code generated, display code  
ELSE display error prompt to try again

|  |  |  |
| --- | --- | --- |
| **Method Name:** CheckItemsRequired | **Class Name:** Inventory Management System | **ID:** 3 |
| **Clients (Chef)** | Main method | |
| **Associated Use Case** | Check for available items | |
| **Description of Responsibilities** | 1. Receive information request from Chef  2. Send request to dining DB 3. Receive info from DB 4. Predict/Calculate item required 5. Display prediction to Chef | |
| **Arguments Received** | Information request | |
| **Type of value returned** | Items prediction | |
| **Pre-conditions** | None | |
| **Post-conditions** | Prediction of quantity of items required | |

**Pseudocode**:  
Receive request from Chef

Send request to Dining DB

Pull up updated database based on current inventory

Receive info from DB  
CALCULATE required items based on shortcoming

RETURN value of prediction/calculation to Chef

|  |  |  |
| --- | --- | --- |
| **Method Name:** CheckItemAvailability | **Class Name:** Inventory Management System | **ID: 4** |
| **Clients (Chef)** | Main method | |
| **Associated Use Case** | Check for available items | |
| **Description of Responsibilities** | 1. Receive availability request from Chef 2. Send request to Inventory DB 3. Receive list of available items 4. Estimate items to be ordered 5. Send estimate list to Manager 6. Send vendor list request to Vendor DB 7. Receive vendor list from Vendor DB 8. Send received vendor list to Manager | |
| **Arguments Received** | Availability request | |
| **Type of value returned** | Estimate of items required and vendor list | |
| **Pre-conditions** | None | |
| **Post-conditions** | Display list of items needed and vendors to order it from | |

**Pseudocode:**

Receive Availability request from Chef

Send request to Inventory DB

Pull up list of inventory items

ESTIMATE required items  
IF Items less than Inventory THEN send flag and vendor list of items

ELSE send all well message

|  |  |  |
| --- | --- | --- |
| **Method Name:** OrderRequiredItem | **Class Name:** Inventory Management System | **ID: 5** |
| **Clients (Manager)** | Main method | |
| **Associated Use Case** | Send approval for quantity required | |
| **Description of Responsibilities** | 1. Receive order request from Inventory Management System 2. Send request to Manager for approval 3. Receive approval or denial 4. Place order if approved 5. Register denial log if denied | |
| **Arguments Received** | Order request | |
| **Type of value returned** | Log of Confirmation of order to vendor or Denial of order | |
| **Pre-conditions** | None | |
| **Post-conditions** | Update DB | |

**Pseudocode:**

Receive order of ingredients with vendor details from system (after prediction)

Send order request to Manager

IF Manager approves, THEN place order  
Update DB for placed order

ELSE

Update DB log for denied order

|  |  |  |
| --- | --- | --- |
| **Method Name:** StockRefill | **Class Name:** Inventory Management System | **ID: 6** |
| **Clients (Chef)** | Main method | |
| **Associated Use Case** | Refill stock | |
| **Description of Responsibilities** | 1. Check container weight every five minutes 2. If container weight below pre-decided value, send refill alarm to waiter | |
| **Arguments Received** | None | |
| **Type of value returned** | Alarm | |
| **Pre-conditions** | None | |
| **Post-conditions** | Signal/alarm, sent to waiter station in inventory | |

**Pseudocode**:

Set timer = 0

FOR timer = 5

Check weight of spoons and forks container

IF WEIGHT < Minimum required WEIGHT THEN RETURN alarm

Reset timer =0

# Project Management Deliverables

## 1. Project Activities

* Meeting to discuss project ideas and picking a good business case
* Allocating work to members - splitting 15 points each.
* Setting up and following individual project activity deadlines
* Providing feedback to other members on their activities
* Compiling final report and presentation

## Allocation of activities to team members

|  |  |
| --- | --- |
| **Task** | **Person-in-charge** |
| Executive summary | Sriram |
| Problem Statement | Sriram |
| Business process model using BPMN | Sriram |
| Context Diagram | Ritu |
| Use Case Diagram | Ritu |
| Use Case Descriptions | Ritu |
| Data Dictionary and Class diagram without methods | Ritu |
| Complete class diagram with methods | Manali |
| Sequence Diagram | Manali |
| Functional specification document | Manali |
| Interface Design | Namratha |
| Database design | Namratha |
| Software design | Dilip |
| Project report deliverables | Dilip |
| Project Management deliverables | Dilip |

## 3. Planned timeline

|  |  |
| --- | --- |
| **Task** | **Planned deadline** |
| Brainstorm project ideas | September 10, 2019 |
| Pick and confirm ONE final project idea | September 16, 2019 |
| Draw out project design, write down problem statement and get professor approval | October 1, 2019 |
| Allocate tasks and get started on first steps | October 5, 2019 |

## 4. Execution timeline

|  |  |
| --- | --- |
| **Task** | **Planned deadline** |
| Executive summary | November 7, 2019 |
| Problem Statement | November 7, 2019 |
| Business process model using BPMN | November 7, 2019 |
| Context Diagram | November 14, 2019 |
| Use Case Diagram | November 14, 2019 |
| Use Case Descriptions | November 14, 2019 |
| Data Dictionary and Class diagram without methods | November 18, 2019 |
| Complete class diagram with methods | November 18, 2019 |
| Sequence Diagram | November 21, 2019 |
| Functional specification document | November 21, 2019 |
| Interface Design | November 28, 2019 |
| Database design | November 28, 2019 |
| Software design | December 4, 2019 |
| Project report deliverables | December 4, 2019 |
| Project Management deliverables | December 5, 2019 |

## 5. Minutes of project meetings (date, time, team members present, topic discussed)

|  |  |
| --- | --- |
| **Meeting Date and Time** | **Minutes** |
| September 2, 2019 11 am | * Come up with various possible ideas in the next meeting. * Formed a Whatsapp group to coordinate better |
| September 10, 2019 1 pm | * Several ideas were presented ranging from book publishing, K-12 education system, to complex telecommunications systems. * Three ideas were picked to discuss with Prof. Raghunathan. |
| September 16, 2019 11 am | * The dining system project idea was approved. * Tasks were allocated to each member. * Timeline for the project was drawn out. |
| October 5, 2019 11 am | * Project design was presented on paper and concept was clarified to everyone. * Feedback was exchanged, questions were raised and answered, roadmap was fixed |
| October 22, 2019 1 pm | * Routine meeting to discuss progress and address concerns if any. * Progress on executive summary, context design and use case diagram was discussed. * Decision was made to keep in touch via email and Whatsapp until next meeting |
| November 14 | * Executive summary and context diagram was discussed and road map was laid out. * Project was on-track so we decided to keep up the routine. |
| November 25 | * Penultimate meeting to ensure everything was on track for final submission. * Exam schedule was discussed but we made sure we are still keeping deliverable deadlines. |
| December 5, 2019 10 am | Final meeting to shoot video and submit final project |

-------------------End--------------------